



下载APP



23 | 技术观：做程序员，技术观为何如此重要？

2020-07-08 臧萌

职场求生攻略

[进入课程 >](#)**讲述：臧萌**

时长 12:03 大小 11.05M



你好，我是臧萌。在这节课里，我想跟你谈谈我的技术观。

首先从技术的角度介绍一下我自己。我从业十四余年，除了担任过几个月的管理职位之外，其他时间我都是做一线的编程工作，并且以此为乐。这十四年来，我做的东西主要是偏产品、平台和框架，支撑上层业务，因此在工作中对代码和技术都是略有要求的。这些年来我也一直在学习技术，有的是出于工作需要，有的是单纯出于兴趣。

技术的重要性当然不需要强调，但是有时候物极必反，会让人迷失在技术中。这么多年来，我总结出这么两点技术观。第一，**吃透技术**。你在工作中，用到的技术必须吃透。二，**需求至上**。技术对你来说只是满足业务和用户需求的**手段**。



这两个观点可谓我的肺腑之言。我喜欢做开发，学技术，但是我并不认可技术至上论，从来都认为技术可以凌驾于需求之上。我和很多人交流过，包括和我一样多年来一直从事一线编程工作的人，以及一线程序员的管理者，他们都很认同我的观点。

谈技术观，首先要和你聊聊软件工程师和技术的关系，这样你可以对这个问题有一个更深入的理解。毕竟，我们就是要和技术打交道的。

用技术把事情做成

程序员的大名是软件工程师，也就是说，我们是做工程的。工程就是要落地，要把一件事情做成，这是最根本的，也是软件工程师价值的体现。不管技术有多牛，只要对事情落地没帮助，那都是空气。公司为什么雇佣我们？这不在于技术本身，而在于我们可以用手中的技术，把问题搞定，让项目落地。

软件开发工程师，其实和厨师一样，一个是用技术写代码做软件，一个是用厨艺做菜品。软件开发的技术也是一样，菜刀就是 Java，炉灶就是数据库，炒锅就是网络，要做到融会贯通地使用，还要不忘初心，把事搞定。

简单说了一下程序员和技术的关系，下面我们再来详细说一说我技术观中的两个方面。

吃透技术

技术是一个程序员安身立命之本。既然是看家的本领，那就必须吃透。那么问题来了，如何吃透呢？

我自己的习惯是，把用到的主要技术，再接着深挖一层。只有深挖一层，才能让我感到安心。我自己定义的“吃透的原则”大致有三条：

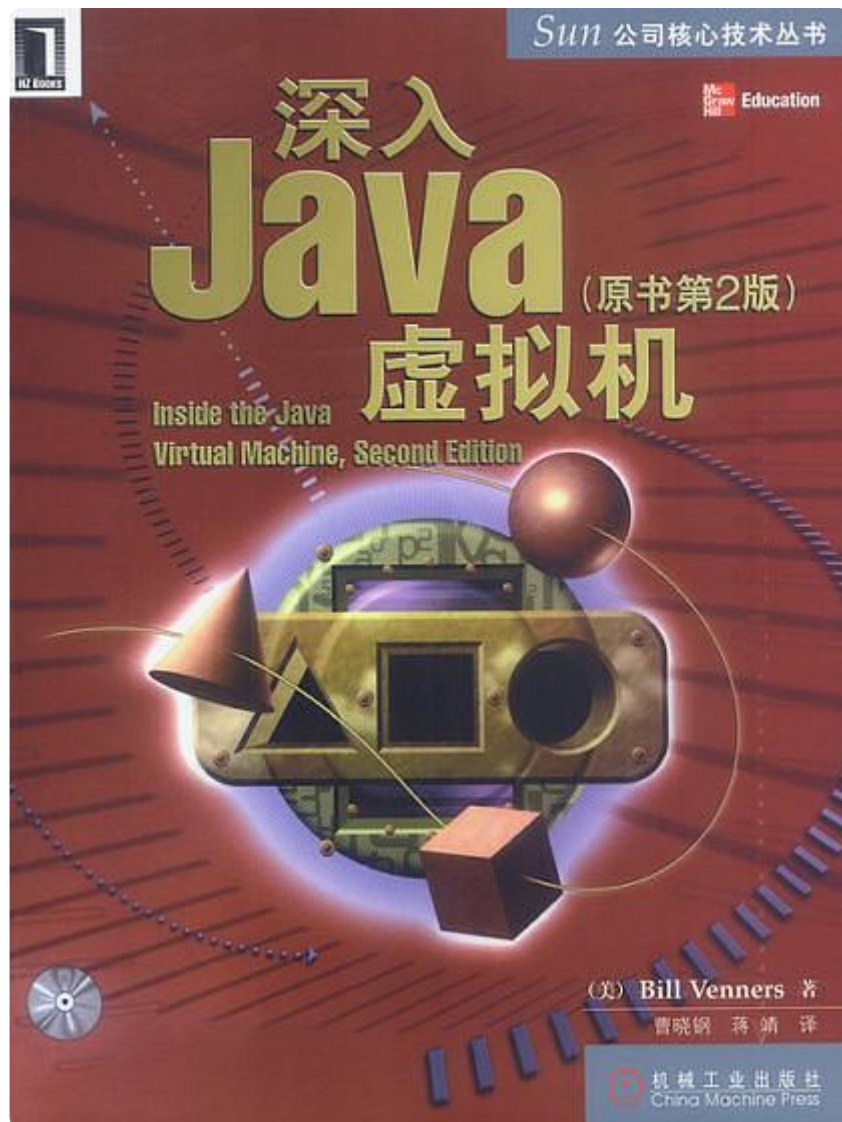
1. 熟练使用：熟练使用技术解决问题，深谙技术的各种坑；
2. 精准掌控：明晰技术的能力边界，准确判断技术能否以及如何满足某个具体的业务场景；
3. 知根知底：知道技术的底层实现和关键细节，如果让自己做，可以大致做出一个来。

怎么理解这三点呢？

举个我自己的例子吧。我大学学的是 Java 语言，大一会写 Java 之后，我确定了两条进阶路径：一手抓“写”，一手抓“学”。

我当时做了一个自己的小网站，做了计算器、拼图游戏、聊天程序、联机对战的俄罗斯方块等等各种杂七杂八的东西。我还熟悉了各种 Java 的类库和工具，比如 JDBC，Tomcat 4，网络，文件，Java Swing/AWT，Applet，Graphics 等等...我开始和各种乱码作斗争，和各种环境与配置做斗争，练就了一双高度近视但是能从万千个 O 中找到 0 的火眼金睛，并乐此不疲。

在那个互联网和电脑刚刚开始普及，Java 在中国还是小众的时候，学习资料的获取还是存在一定限制的，主要靠学校图书馆。就在我学得顺风顺水的时候，我遇到了这本书《深入 Java 虚拟机（原书第二版）》：



说实话，我从图书馆借来的书，看不懂的有很多。什么 EJB 之类的，我感觉跨度太大，就弃坑了。但是这本书，我觉得我应该要看懂，这就是我要深挖的那一层——我要知道 Java

是怎么跑的。

那个年代学习并没有这么便捷，一切只能靠自己硬啃。于是整个大二，我借了还，还了借，一年下来，反反复复地看，只能算是勉强看了 3 章。这本书有 461 页，不包含附录有 21 章。这本书是我大二的痛苦回忆。

好在啃过前几章之后，也可能是后面的内容简单了，也可能是自己写得多了悟到了。大学毕业前，我把这本书基本啃完了，这才算是对 Java 有了一个系统的理解。

工作以后，才算是踏上了吃透 Java 的漫漫旅程。工作之初，可以说是看啥啥不懂，用啥啥不会。这时候算是开荒，疯狂学习各种框架、类库、工具。紧接着，我工作中遇到了各种千奇百怪的问题。我的态度是，能绕过就绕过，先解决问题。这样的目的是能够在不背着业务压力，不受时间约束的情况下，有充裕的时间去找出问题原因。

程序出问题，肯定有原因。我坚信这一点，需要挖多深就挖多深，是为了避免以后再糊糊涂涂地趟进同一个坑。不放过 log 里任何一个无法解释的异常，不放过任何一个看不透的细节。当然，这里学的有些东西，并不是为了使用，而是为了知其不可为而不为之，或者是为了不被不知其不可为而为之的人坑，是不是有点绕，哈哈。

最后，我再结合这段经历做个总结，让你更直观地感受一下各个阶段的时间线以及需要做的工作。

1. 熟练使用：需要做到只要面对 Java 的问题，你都不觉得难。
2. 精准掌控：知道 Java 和 Java 生态的长处与短处。看到一个需求，直觉判断出可行性，难点以及可能出问题的地方。
3. 知根知底：以 JVM 为分割线，熟悉其上的类加载器、字节码、内存分配和内存模型这些 Java 生态的基石。这样做一方面是为了你能够进一步精准掌握 Java。另一方面，Java 作为一个设计优秀的语言，在设计自己的 DSL 的时候很多地方是值得参考的。

我面试时，如果看到有同学的简历上写着“精通某项技术”，我就会问一个标准问题：“使用这个技术的时候你有遇到过什么问题，踩过什么坑吗？”如果这个同学的回答非常干瘪，那么就说明面试者只做到了第一点，也就是熟练使用。因为如果深入地长期使用一个技术，肯定会有业务需求和技术能力边界的各种摩擦和冲突，踩过大大小小的各种

坑。当然没有遇到问题自然是好的，那只能说明这个面试者很幸运。但是少了挫折，也让人距离精通这门技术，差了点火候。

需求至上

需求至上，是我的第二个技术观。需求一般来自于用户和业务。

我这里说的用户，不是最终用户，而是我们做的软件系统的用户。用户可能是同一个公司的不同部门，也可能是对口的业务公司。绝大多数情况下，用户非常明确地知道自己想要什么，非常明确地知道自己想要解决的问题。

那么技术到底如何支撑业务呢？

我的看法是，多和用户以及业务方交流，理解他们的需求，才能更好地施展技术。而不能本末倒置，自己想做成什么样，就做成什么样。业务方不理解，那是他们 low；用户不会用，那是他们笨。如果抱着这种态度，那么用再好的技术也做不成事情。

但沟通自然没那么简单。

刚开始你和业务方以及用户交流的时候，会有鸡同鸭讲的感觉，但是请相信，随着交流的深入，观点的碰撞，作为程序员的我们，会更能够理解自己所做事情的价值，也更能够选择正确的技术和解决方案。

这也是为什么很多理论派、技术流的产品，很难获得市场。但是很多实干派的产品，却可以赢得市场。技术流的产品，做出来一般都很酷，但是却把用户的需求和实际的业务问题放在了一边。而实干派，就是立足于解决用户的问题，满足业务需求，所以更容易赢得市场的认可。

举个例子，为什么 Google 在云计算市场战不过亚马逊？在我看来，Google 由于太过于技术流而脱离了需求。

亚马逊在 2006 年推出了 AWS，最开始的产品是 IaaS、S3。当时 IaaS 尚属于新事物，但是和传统的虚拟主机差不多，IaaS 的易用性和扩展性，大大地方便了很多公司，满足了他们对于计算和存储的需求。

Google 在 2008 年推出了 Google App Engine (简称 GAE) 。而 Google 这一步，迈得有点大。在 GAE 上写程序，必须用 Google 自己的 SDK，存储要用 Google 自己的 KV 数据库，看起来很炫酷，但是这些炫酷，甚至可以称为是“划时代”的东西，能赢得技术拥趸的叫好，却很难赢得公司的叫座。为什么？因为不实用嘛。

亚马逊胜就胜在“实用”二字。各种方便实用的功能，都做到用户心坎里去了。监控，报表，工具怎么用怎么顺手，怎么玩怎么顺心。而 GAE，用“华而不实”形容并不为过。回过味之后，Google 现在也回来好好做 IaaS 了，但是基因难改，各种边边角角的地方，都能闻出华而不实的技术派味道。

还有很多小例子。比如一个公司，按照自己理解的业务方向，去解决自己认为有价值的问题。公司产品做了三年，拿出来给用户一看，用户说压根就不是他们想要的样子。比如更小的事情，各种监控功能做得花里胡哨，但是对于用户最关心的销量指标 week by week 对比，却要用户开两个浏览器去人肉比。

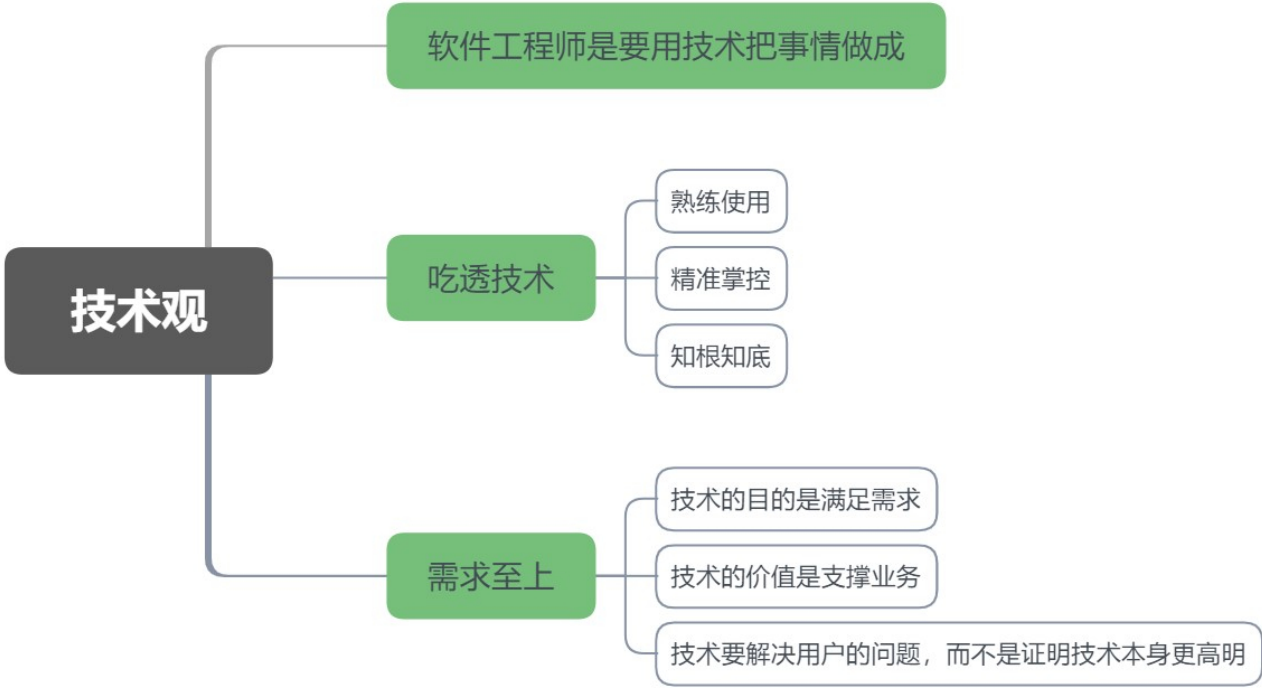
这不是逗呢嘛？

做技术一定要懂业务，也就是要理解我们要解决的问题。你一定要懂得，怎么用技术为业务创造价值，而不是给业务添乱。

总结

我们除了要死磕技术之外，还要理解技术所服务的业务领域，理解用户的需求。这样，才不至于在死磕中，碰晕了头，犯想当然的错误。双管齐下，才能进一步理解自己工作的价值，朝着有价值的方向前进。

对待技术，我们不妨换个角度，不要认为自己是在学习技术，而应该这样认为：日常的工作都是在锻炼解决问题，而非技术。技术只是顺带学的。工程师的核心能力，是理解问题和解决问题的能力。



思考题

不知道你看完这篇文章有什么感想呢？你觉得你吃透技术了吗？对于你现在常用的技术，如果让你自己去实现，你有思路吗？你理解自己所做的系统的需求吗？

好，今天的课程就结束了，希望可以帮助到你，也希望你在下方的留言区和我参与讨论，并把文章分享给你的朋友或者同事，一起交流一下。

提建议

更多课程推荐

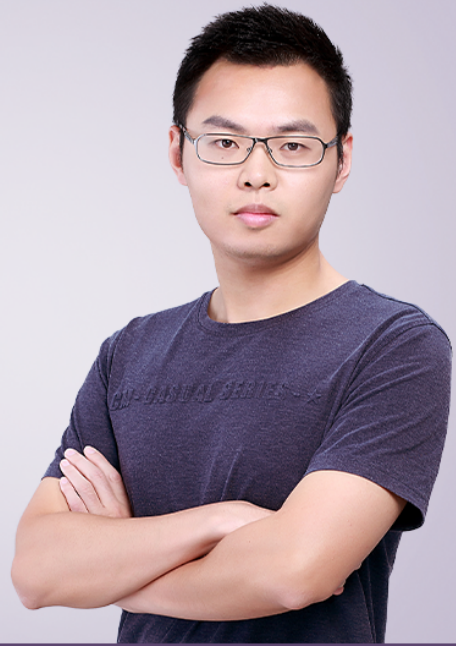
设计模式之美

前 Google 工程师手把手教你写高质量代码

王争

前 Google 工程师

《数据结构与算法之美》专栏作者



涨价倒计时 🕒

限时秒杀 **¥149**，7月31日涨价至 **¥299**

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 22 | 学习观：程序员如何定义自己的技术舒适区？

下一篇 24 | 技术观：程序员在技术的成长之路上，有哪些陷阱？

精选留言 (4)

写留言



Sdylan

2020-07-08

空谈技术，不顾业务，只会误了技术误了业务。技术本来就是为了解决问题而生的，若脱了问题。就会变得毫无价值，生命周期也是短的。只有落地，随着业务发展才能长久。对个人而言，用到必须里外吃透，做的需求至少要达到为啥做需求，解决什么业务痛点。

展开 ∨

作者回复: 其实做技术多了会发现，技术过几年大概率会扔掉，而业务反而不容易扔掉，积累起来越来越值钱。

其实我们程序员容易忽视业务，或者说容易钻到技术里出不来，也是有客观原因的。就是做技术需要全心全意的投入，就是需要你钻进去。这时候，就容易忽略我们的初心：服务业务。所以我

们程序员可以时不时的从技术的世界里钻出来，重新想想业务是什么，我们钻的深度和方向，是否还是为了业务服务。



🌸🌸🌸人...

2020-07-09

我又来了，一直纠结几个问题。

应届生刚出来所在公司小组的方向：做业务和做技术的优劣在哪里？

两个方向如何进行选择？

如果现在的小组方向就是做业务了，如何在工作中增强自己的能力和技术？...

展开 ∨

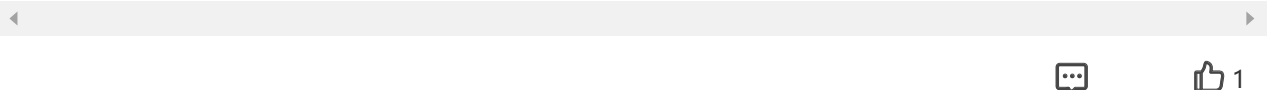
作者回复: 应届生刚出来所在公司小组的方向：做业务和做技术的优劣在哪里？

这个取决于不同的人。如果你觉得自己对技术的兴趣更大，就可以专注于技术。

如果你觉得业务对你来说更有吸引力，就可以考虑走业务方向。

方向本身没有优劣，也没有严格的分界线。做技术也要懂业务，做业务也要懂技术。也就是说，都要到合格线。选方向是上线，不是下限。

如果是专注业务方向的话，写的代码主要是业务代码。其实对技术的要求不会太高，有精力的话，可以试着学习一下业务使用的框架，这会对业务开发有帮助。



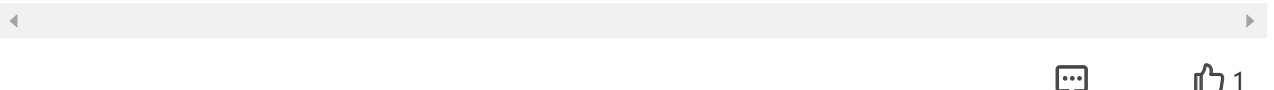
Middleware

2020-07-08

软件工程师依靠自己的技术最终可落地才是硬道理。

展开 ∨

作者回复: ✓，👍，就是这么个意思。勿让初心。



白酒小林

2020-07-08

代码值不值钱还是要靠产品决定。

展开

作者回复: 对，面粉值多少钱得看是坐馒头还是蛋糕



1